VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2014

Peter Slivoš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Peter Slivoš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: AstrumQ Iteractive, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

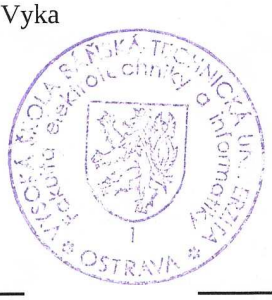
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Kot, Ph.D.**

Konzultant bakalářské práce: Bc. Aleš Vyka

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 24. 3 2014

Za pomoc pri práci a rady v každej situácií chcem poďakovať hlavne môjmu skvelému mentorovi Radimovi Halfarovi. Následne chcem poďakovať aj Alešovi Vykovi a jeho firme AstrumQ za pracovnú príležitosť.

Abstrakt

V tejto bakalárskej práci opisujem priebeh mojej bakalárskej praxe, ktorá sa uskutočnila vo firme AstrumQ Interactive s.r.o.. Popisuje software, ktorý som počas praxe vyvíjal, problémy s ktorými som sa počas jeho vývoja stretol a moje riešenie daného problému. Cieľový software je mobilná aplikácia na platforme Android, ktorá má za úlohu zjednodušiť predaj zákazníckej firme. Taktiež opisujem moje celkové dojmy z praxe a nadobudnuté skúsenosti.

Kľúčové slová: mobilná aplikácia, platforma, Java, Android, software, vývoj, technológia, návrhový vzor, bakalárska práca, Eclipse, GIT, JVM, JRE, ADT

Abstract

In this bachelor thesis I describe my work during bachelor practice, which was carried out in AstrumQ Interactive s.r.o. It describes the software which I have developed during this practice, problems related to the work as well as solutions to those problems. Final software is an Android mobile application, which purpose is to ease the marketing of a given company. I also describe my personal feelings and experiences gained from this practice.

Keywords: mobile application, platform, Java, Android, software, developement, design patterns, bachelor thesis, Eclipse, GIT, JVM, JRE, ADT

Zoznam použitých skratiek a symbolov

JVM	–	Java Virtual Machine
JRE	–	Java Runtime Enviroment
ADT	–	Android Developer Tools
GIT	–	GNU Interactive Tools
GNU	–	GNU is Not Unix
IDE	–	Integrated Development Environment
GPS	–	Global Positioning System

Obsah

Zoznam obrázkov

Úvod

Bakalársku prax som si vybral hlavne preto, že som chcel zistiť, ako porogramovanie funguje vo firmách. Chcel som sa tým teda hlavne pripraviť na svoje budúce zamestnanie. Prácu v kolektíve som si dovtedy taktiež v škole moc nevyskúšal.

Hlavne tieto dôvody ma viedli k navštíveniu školského workshopu, na ktorom sa prezentujú firmy ponúkajúce prax. Najviac ma zaujala firma AstrumQ, ktorá sa venuje webovým riešeniam, mobilným aplikáciám a internetovému marketingu. Hľadali programátorov, ktorí ovládajú jazyky Java a .Net. Keďže mi škola dala výborný základ pre oba tieto jazyky, tak som sa u nich mohol prihlásiť na prax. Na prijímacom pohovore som sa dozvedel, že zadaním mojej práce je vývoj mobilnej aplikácie pre platformu Android (programovací jazyk Java). Zadaná téma sa mi pozdávala a tak som sa s firmou dohodol na podmienkach praxi.

V práci popíšem moje začiatky vo firme, prácu v kolektíve , štruktúru vyvíjanej mobilnej aplikácie a jej funkčnosť. Zároveň popíšem niektoré problémy a k nim moje riešenie.

Keďže firma, v ktorej som získal bakalársku prax, ma viaže zmluvou o mlčanlivosti, ktorá zabraňuje úniku konkrétnych dát a know how postupov, musím túto prácu písať bez konkrétnych faktov a príkladových zdrojových kódov. Napriek tomu sa ale pokúsim, aby boli fakty a riešenia ohľadom aplikácie čo najzrozumiteľnejšie.

1 Prvé dni na pracovisku

Na začiatku mojej praxe mi bol pridelený mentor, ktorý ma oboznámil s detailami práce a previedol po pracovisku. Zoznámil ma s niektorými ostatnými zamestnancami a vysvetlil mi ich role v kolektíve. Spoločne sme sa dohodli na pracovnom rozvrhu a dochádzke.

1.1 Pracovisko

Pracovisko sa nachádza v areáli školy, čo bola pre mňa veľká výhoda. Samotné pracovisko je dobre technicky vybavené. Majú tam napríklad mobilné zariadenia, na ktorých je možné vyrobené aplikácie testovať. Taktiež sa tam nachádza flipchart, na ktorom mi mentor znázorňoval programovacie postupy, vďaka ktorým sa vždy programovalo oveľa lahšie. Z praktických dôvodov som si tam nosil vlastný notebook a mal som svoje vlastné miesto na prácu vo firme. Firma má samozrejme aj bezdrôtové pripojenie na internet a množstvo voľných elektrických zásuviek. Vo firme tiež vládne pozitívna tímová nálada a všetci zamestnanci si ochotne pomáhajú.

1.2 Firemný informačný systém

Mentor mi vytvoril účet vo firemnom systéme, v ktorom firma eviduje svoje pracovné úlohy a prideľujú k nim strávený čas nad konkrétnym problémom. Taktiež sa v ňom nachádzajú aj dokumentácie ku konkrétnym úlohám. Systém som si skúsil na príkladnej úlohe. Jeho obsluha bola jednoduchá a rýchla.

1.3 Príprava softvéru

Po oboznámení so systémom som dostal zoznam softvéru, ktorý potrebujem na vývoj aplikácie. Patrí do neho Java Runtime Enviroment, vývojové prostredie Eclipse, Android Developer Tools plugin do programu Eclipse a GIT.

2 Použitý software

2.1 Java

Java[3] je počítačový, objektovo orientovaný programovací jazyk. Je založený na triedach, ktoré môžu reprezentovať objekty v reálnom svete. Je robustný a bezpečný, čo znamená že sa v ňom dajú písať veľké stabilné aplikácie. Taktiež podporuje použitie procesových vlákien. Programy napísane v tomto jazyku sú spustiteľné na každom počítači, ktorý má nainštalovaný Java Virtual Machine.

2.1.1 Java Runtime Enviroment (JRE)

Java Runtime Enviroment (JRE) je súčasťou Java Developer Kit (JDK), ktorý je súhrnom programátorských nástrojov pre vývoj Java aplikácií. Java Runtime Enviroment poskytuje potrebné minimum pre spúštanie Java aplikácií a je nevyhnutné na používanie akejkoľvek Java aplikácie. JRE tvorí Java Virtual Machine (JVM), jadrové Java triedy a podporné Java knižnice.

2.1.2 Java Virtual Machine (JVM)

Java Virtual Machine je sada programov a dátových štruktúr. Využíva modul virtuálneho stroja ku spusteniu dalších Java programov. Java Virtual Machine dokáže spracovať Java bytecode, ktorý je obvykle vytvorený zo zdrojového kódu Java aplikácií. Taktiež umožňuje automatické spracovávanie výnimiek, pri ktorých dokáže nájsť zdroj problému bez ohľadu na zdrojový kód.

2.2 Android

Android[2] je operačný systém, ktorého jadro je postavené na kernele Linuxu. Je vyvinutý primárne pre zariadenia s dotykovými obrazovkami. Ovládanie Android aplikácie prebieha na základe dotykových vstupov. Väčšina Android hardwaru obsahuje akcelerometre, gyroskopy a senzory blízkosti, ktoré reagujú na niektoré špecifické akcie.

Vyvíjanie aplikácie pre Android som si vybral hlavne preto, že sa mi tento operačný systém páči a že ovládam programovací jazyk Java. Aplikácie majú jednoduchý ale premyslený design, systém je spoľahlivý a interakcia s aplikáciami je intuitívna. Ďalší dôvod je aj fakt, že sa Android komunita v dnešnej dobe prudko rozrastá a dopyt na trhu je vysoký.

2.3 Eclipse IDE

Eclipse je prográmátorské vývojové prostredie určené primárne pre Java platformu. Obsahuje systém rozšírení, vďaka ktorému si programátor môže plne upravovať svoje vývojvé prostredie. Pomocou týchto vylepšení sa môže Eclipse používať aj na iné programovacie jazyky. Jeho hlavná výhoda spočíva v spustiteľnosti na každom systéme, ktorý

má v sebe Java Virtual Machine a v jeho jednoduchej rozšíriteľnosti. Každá funkcia okrem jeho jadra sa správa ako rozšírenie samotnej aplikácie.

Jedno z hlavných rozšírení, ktoré som používal ja, je LogCat. LogCat vypisuje všetky procesy aplikácie do svojho logu. Do tohto logu je však možné robiť výpisy z akejkoľvek časti zdrojového kódu. Výpisy majú viac druhov a každý druh má svoju unikátnu farbu. Keďže som si mohol vypísať akúkoľvek premennú do logu a zistiť problém okamžite, ušetril som čas, ktorý by som inak strávil debugovaním.

2.3.1 Android Developer Tools (ADT)

Android Developer Tools je nadstavba pre vývojové prostredie Eclipse určená pre vývoj Android aplikácií. Obsahuje Android knižnice, grafické rozhranie prispôsobené mobilným aplikáciám a emulátor, v ktorom sa vyvíjané aplikácie môžu testovať.

Užívateľské rozhranie ADT poskytuje všetky výhody, ktoré poskytuje Eclipse, s úpravami, ktoré sedia pre vývoj Android aplikácií. Vylepšuje pôvodné XML editory, obsahuje predrobené šablóny pre štandardné Android aplikácie a obsahuje upravený refractoring kompatibilný s Androidom. Podporuje aj pripojenie reálneho mobilného zariadenia cez USB k počítaču, čo veľmi uľahčuje testovanie. V neskorších fázach vývoja ale nebolo možné vstavaný emulátor používať. Knižnice Google maps musia byť spúšťané na reálnom zariadení.

2.4 GNU Interactive Tools (GIT)

GIT je distribuovaný systém správy verzií a manažér zdrojového kódu. Dokáže zálohovať a verzovať zdrojové kódy na firemnom serveri. Zdrojový kód sa vždy po vypracovaní nejakého funkčného celku zašle na server, čím sa predchádza akejkoľvek strate dát. Veľké využitie má aj pri obnovení zdrojového kódu do predošlého funkčného stavu.

Na server som vždy posielal zdrojový kód, ktorý neobsahoval žiadne kompilačné chyby. Niekoľkokrát mi GIT zachránil zdrojový kód, keď sa mi ho podarilo nenávratne znefunkčniť. Spätné prepisovanie kódu by mi zabralo mnohonásobne viac času ako jednoduchý návrat do posledného funkčného bodu programu. Zároveň mi niekedy na GIT poslal mentor môj zdrojový kód s jeho poznámkami. Vďaka tomu bola implementácia efektívnejšia.

2.5 Adobe Photoshop

Adobe Photoshop je grafický program, primárne určený na úpravu bitmapových obrázkov. Obsahuje veľmi veľké množstvo rôznych nástrojov, čo z neho robí jednotku na trhu. Tento program som využil na orezávanie obrázkov, ktoré som dostal od firemného grafika. Tie boli použité ako súčasť aplikácie.

2.6 Rozdiely medzi Android a Java platformami

Hoci je väčšina Android aplikácií napísaných v Jave, Android a Java sa líšia v mnohých aspektoch. Android miesto Java Virtual Machine využíva vlastný virtuálny stroj Dalvik. Na rozdiel od Javy, ktorá má architektúru zásobníkových strojov, je Dalvik architektúra založená na registroch. Dalvik bol tiež navrhnutý tak, aby spotrebovával menej miesta ako virtuálny stroj. Ďalší rozdiel je, že sa pri Android aplikáciach nespúšta Java bytecode, ale triedy sú konvertované do .dex (Dalvik EXecutable) alebo .odex (Optimized Dalvik Executable) formátu.

Jeden z praktických rozdielov v zdrojovom kóde je napríklad AndroidManifest.xml. Do tohto súboru sa musia zapísať všetky obrazovky, ktoré aplikácia obsahuje, inak nemôžu byť spustené.

Zrejmý rozdiel je aj vo vytváraní grafiky a typoch rozmiestnení prvkov v obrazovke. Java rozmiestnenia nie sú stavané na dotykové obrazovky a preto obsahujú iné prvky. Android obsahuje menej typov rozmiestnení prvkov v obrazovke a preto mi prišlo robenie grafiky trochu obtiažnejšie ako v Java alikácií. Natívne dotykové prvky už ale majú vstavané všetky potrebné operácie, na ich ovládanie dotykom. Jediný problém, ktorý ma trápil najviac, bola rýchlosť emulátoru, na ktorom som aplikáciu testoval. Reakčný čas a animácie boli v norme, ale načítanie samotného emulátoru trvalo v priemere aj viac ako 6 minút.

3 Aplikácia

3.1 Zmysel aplikácie

Ceľová aplikácia je určená pre konkrétnu firmu zaoberajúcu sa telekomunikačnými službami v oblasti pevných liniek, mobilného volania a internetu. Má slúžiť ako rozhranie medzi zákazníkmi a firmou, ktorá svojim zákazníkom ponúka svoje výhody. Potenciálny zákazník si môže tieto výhody prezerať, filtrovať a zobrazovať si ich na mape. Po prihlásení má možnosť konkrétne výhody zakúpiť.

Hlavnou úlohou aplikácie je uľahčiť predaj firme, ktorá ponúka svoje výhody. Zároveň slúži aj ako náhrada firemnej webovej stránky. Aplikácia je určená na platformu Android, ale firma si ju objednala aj na platformách iOS a Widows Phone. Na ostatných platformách pracovali ale iní zamestnanci.

Pri práci na aplikácii som nadobudol veľa programátorských schopností. Vďaka tejto praxi som teraz schopný napísať vlastnú Android aplikáciu.

3.2 Architektúra aplikácie

Aplikácia je postavená na softvérovej architektúre MVC (Model-View-Controller). Pri jej vyvýjaní som použil niekoľko návrhových vzorov, ktoré sa obecne používajú v praxi. Pri vyvýjaní som využil externú knižnicu Google maps v2.

4 Návrhové vzory

Návrhové vzory[1] predstavujú obecné riešenie problému. Slúžia ako šablóna, ktorá môže byť použitá v rôznych situáciách. Objektovo orientované návrhové vzory typicky ukazujú vzťahy a vzájomné pôsobenie medzi triedami a objektami. Existuje viacero typov návrhových vzorov. Najznámejšie sú vzory týkajúce sa tvorby objektov, štruktúry programov a chovania systému.

4.1 Vzory týkajúce sa tvorby objektov

Tieto návrhové vzory riešia problematiku okolo vytvárania objektov v systéme. Popisujú postup výberu triedy nového objektu a zaisťujú správny počet týchto objektov.

4.1.1 Singleton

Návrhový vzor Singleton zaisťuje to, aby mala daná trieda iba jednu instanciu. Využil som ho niekoľkokrát. Napríklad trieda, ktorá obsluhuje vlákno bežiace na pozadí aplikácie, môže mať iba jednu instanciu. Nemá význam mať viac instancií, pretože server vždy rieši iba jeden požiadavok naraz a jedno vlákno na dané požiadavky úplne stačí. Keď sa jeden požiadavok vykoná a zo servera sa vráti správa, správa sa spracuje a vlákno je znova použitelné.

4.2 Vzory týkajúce sa štruktúry programov

Tieto návrhové vzory sa zameriavajú na usporiadanie a sprehľadnenie jednotlivých tried alebo rôznych komponentov stystému. V projekte som nevyužil žiadny z týchto návrhových vzorov.

4.3 Vzory týkajúce sa chovania systému

Tieto návrhové vzory môžu byť založené na triedach alebo objektoch. U tried sa týkajú hlavne dedičnosti a pri objektoch sa rieši vzájomná spolupráca medzi objektami alebo skupinami objektov.

4.3.1 Observer

Viac objektov, ktoré sú závislé na jednom objekte, budú všetky notifikované, keď sa daný objekt zmení alebo vyšle signál notifikácie. Tento návrhový vzor využívam viacnásobne. Zakaždým, keď sa vytvára požiadavok na server a vráti sa správa, trieda ktorá spracuje túto správu vyšle notifikáciu všetkým potrebným triedam. Niektoré triedy len potrebuju vykonať jednoduchú operáciu, ako napríklad zavretie okna pri prihlásení do systému, zatiaľ čo iná trieda obdrží z tej istej notifikácie dáta o prihlásenom užívateľovi.

5 Architektúra Model View Controller

Táto architektúra oddeľuje dátový model aplikácie, užívateľské rozhranie a riadiacu logiku do nezávislých komponentov tak, že ich modifikácia má minimálny vplyv na ostatné komponenty. Tieto nezávyslé komponenty sú Model View a Controller vrstva.

Bežným programátorským postupom je zoradiť si súbory v projekte hierarchicky podľa týchto troch vrstiev. Projekt je potom omnoho prehľadnejší.

Model vrstva reprezentuje dáta v aplikácii a oboznamuje View a Controller vrstvy vždy, keď v týchto dátach nastane akákoľvek zmena. Toto oboznámenie umožní View vrstve obnoviť svoj stav a Controller vrtva môže podľa potreby zmeniť logiku aplikácie. V niektorých prípadoch môže byť Model vrstva pasívna. To znamená, že View a Controller vrstvy si žiadajú o oboznámenie od Model vrstvy sami.

View vrstva prijíma informácie z Controller vrstvy, ktoré slúžia na to, aby mohla View vrstva vykresliť užívateľské rozhranie. Taktiež môže informovať Controller vrstvu o vstupoch, ktoré užívateľ aplikácie zadal.

Controller vrstva môže Model vrstve poslať príkaz na obnovenie stavu Model vrstvy. Má za úlohu posielať aj príkazy View vrstve, aby zmenila svoju reprezentáciu Model vrstvy. Príklad tejto akcie je scrollovanie dokumentu.

5.1 Model

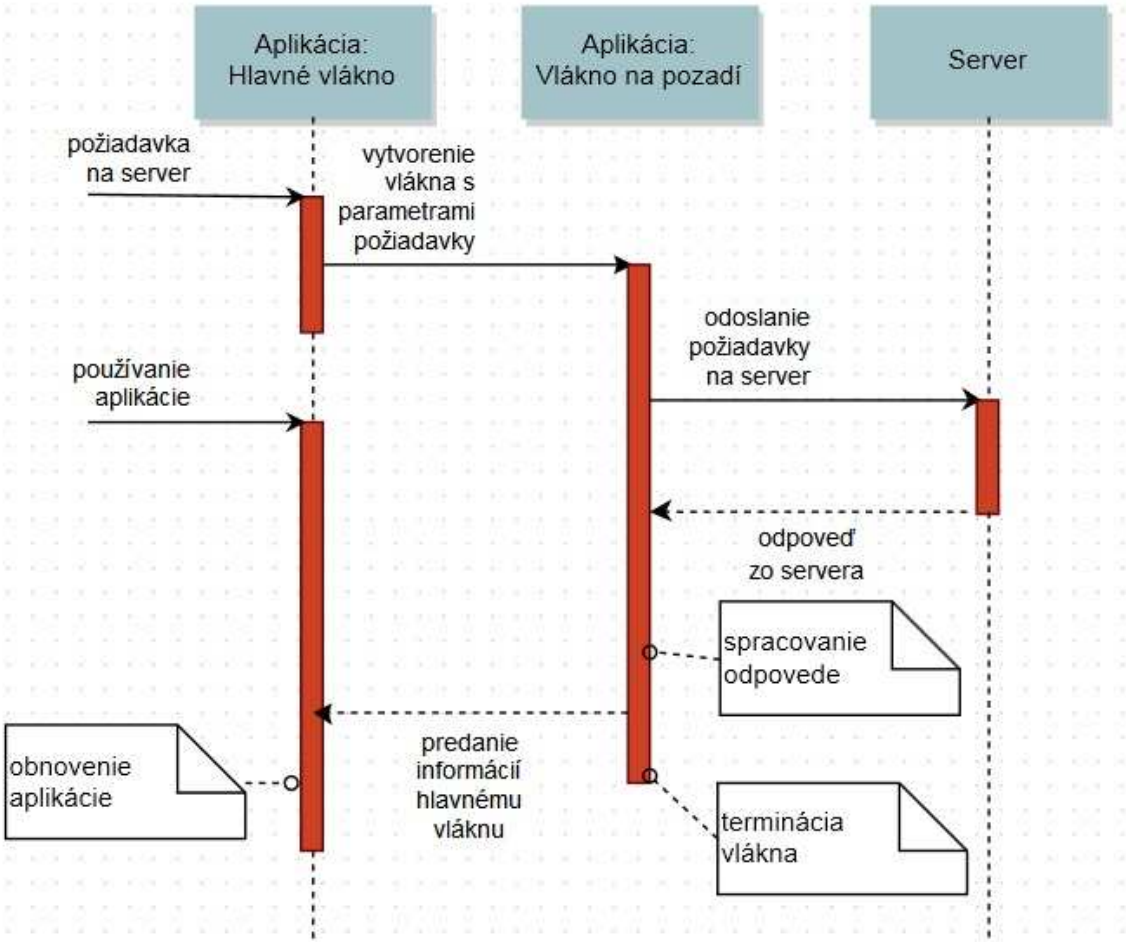
Moja Model vrstva obsahuje triedy, ktoré obsluhujú komunikáciu so serverom, konštanty, dočasné dáta a adaptéry.

5.1.1 Komunikácia so serverom

Komunikácia so serverom v Android aplikáciách musí prebiehať asynchrónne. Vlákno bežiace na pozadí sa stará o posielanie požiadavku na server, odpoveď zo servera a následovné spracovanie dát zo servera. Keby tomu tak nebolo, celá aplikácia by stála a čakala by na odpoveď zo servera. To môže trvať niekedy dlhší čas. V niektorých hraničných prípadoch dokonca odpoveď zo servera ani nemusí doraziť. Keď sú však tieto požiadavky na server vykonávané pomocou vlákna na pozadí, užívateľ môže voľne využívať aplikáciu a nič si nevšimne.

Kvôli tomuto aplikácia obsahuje triedu určenú pre komunikáciu so serverom, ktorá dokáže odosielať konkrétne požiadavy na server s určitými parametrami cez asynchrónne vlákno na pozadí. Požiadavky sú posielané v textovej forme s využitím značkovacieho jazyku. Parametre jednej požiadavky sú špecifické pre každú požiadavku. Každá požiadavka odoslaná na server obdrží zo servera špecifickú odpoveď. Preto je pre každú konkrétnu požiadavku vytvorená obslužná trieda menom Handler, ktorá danú požiadavku dokáže spracovať a následne predať získané dáta zo servera ďalej do aplikácie. Spracovanie odpovede prebieha pomocou parsovania a následného napĺňania konkrétnych objektov naparsovanými hodnotami.

Každá požiadavka v aplikácii určená serveru musí prebiehať týmto spôsobom. Vlákno odošle požiadavku na server a čaká na odpoveď. Keď odpoveď zo servera príde, správa sa spracuje a vlákno zanikne.



Obr. 1: Interakcia vlákien so serverom

5.1.2 Konštanty

Medzi konštanty patria napríklad hodnoty, ktoré sa využívajú na určenie adresy požiadavky pre server. Adresa požiadavky sa skladá z viacerých častí. Taktiež sa tu nachádzajú aj jazykové balíčky, z ktorých aplikácia čerpá všetky textove prvky.

5.1.3 Dočasné dáta

Dočasné dáta sú všetky dáta, ktoré aplikácia používa pri svojom chode. Medzi ne patria údaje o prihlásenom užívateľovi, list firemných výhod, list kategórií a list lokácií. S týmito dátami pracuje aplikácia iba počas jej behu. Pri vypnutí aplikácie sú zmazané.

5.1.4 Uložené dáta

Medzi tieto dáta patria prihlasovacie údaje užívateľa. Ukladajú sa iba vtedy, keď užívateľ chce, aby si aplikácia jeho údaje zapamätala. Ukladajú sa do pamäte Android zariadenia a predtým ako sú uložené, sa z bezpečnostných dôvodov šifrujú.

5.2 View

Moja View vrstva obsahuje súbory, ktoré definujú vzhľad aplikácie v značkovacom jazyku. Každá obrazovka má vlastný súbor, v ktorom sú definované jednotlivé prvky. Prvky objektov, ktoré sa opakujú, ako napríklad položky menu alebo firemné výhody, majú všetky rovnaký vzhľad a sú definované len raz. Konkrétne adaptéry sa starajú o to, aby zobrazenie prvkov bolo pre každý prvok správne a unikátne, aj keď grafické rozloženie prvkov je rovnaké.

5.3 Controller

Moja Controller vrstva obsahuje triedy, ktoré sú naviazané na súbory vrstvy view a adaptéry. Predstavujú logiku aplikácie a funkcionalitu konkrétnych vizuálnych prvkov. Do tejto vrstvy patria aj adaptéry.

5.3.1 Logika

Každá trieda tohto typu potrebuje pri vytvorení načítať prvky z vrstvy View, aby bolo prepojenie kompletné. Následne je možné do týchto prvkov načítať konrétne dáta z Model vrstvy.

5.3.2 Adaptéry

Adaptéry slúžia ako prepojovacie body medzi konkrétnymi súbormi vrstvy View a triedou z Model vrstvy. Každý adaptér má definovaný vlastný view súbor, ktorý považuje za jeden prvok z listu objektov rovnakého typu. Jeho hlavná funkcia je naviazať viacero instancií určitej triedy na grafickú časť aplikácie. Vďaka adaptéru sa všetky prvky v liste správajú a vyzerajú rovnako, no sú naplnené inými dátami. Adaptéry dokážu interpretovať viac prvkov rovnakého typu a grafického štýlu s rozdielnymi parametrami pre každý prvok.

Adaptéry som použil napríklad na prvky v oboch vysúvacích menu a na list firemných výhod. Všetky prvky menu zdieľajú rovnaký grafický súbor vo View vrstve a patričný adaptér v Controller vrstve. Dáta z Model vrstvy majú štruktúru listu a tento konkrétny

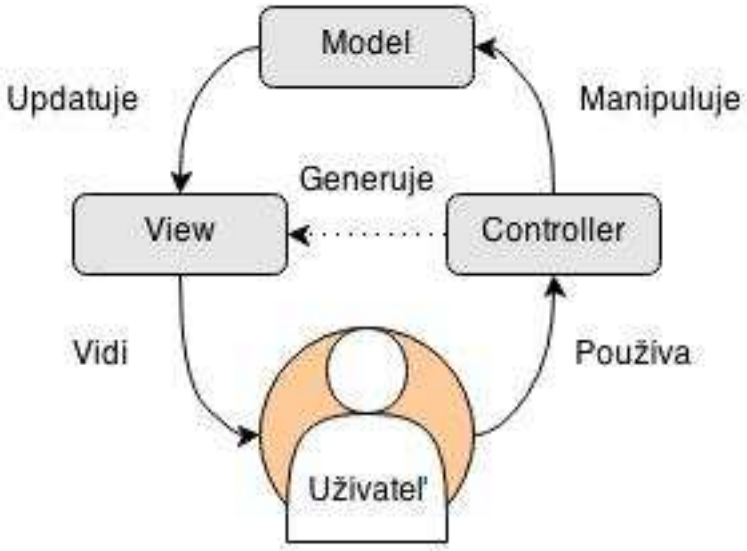
adaptér, starajúci sa o prvky menu, obsahuje tento list prvkov a metódu na vytvorenie jedého prvku po grafickej stránke. Zakaždým, keď potrebujeme tento list obnoviť, zavolá sa na adaptér metóda, ktorá prvky znovu načíta. Trieda z Model vrstvy je nevyhnutná pre adaptér, pretože definuje typ listu a obsahuje všetky dáta, s ktorými adaptér pracuje.

5.4 Komunikácia medzi vrstvami MVC

Táto komunikácia prebieha na základe návrhového vzoru Observer. Využíva sa tu pozorovaný objekt a list objektov, kde každý z týchto objektov je pozorovateľ. V aplikácii je použitý jeden typ pozorovateľa na jeden typ serverovej požiadavky.

Zakaždým, keď sa spúšťa nejaká obrazovka aplikácie (View), sa zaregistrujú pozorovatelia potrebného typu do svojeho listu (Controller). Keď sa daná požiadavka na server vykoná, obslužná trieda, ktorá správu spracuje (Model), oznámi zmenu všetkým svojim pozorovateľom (Controller). Správa, ktorou dá trieda svojim poslucháčom vedieť konkrétnu zmenu, môže niesť akúkoľvek dátovú štruktúru. Následne je možné prijaté dáta zobraziť na obrazovke (View).

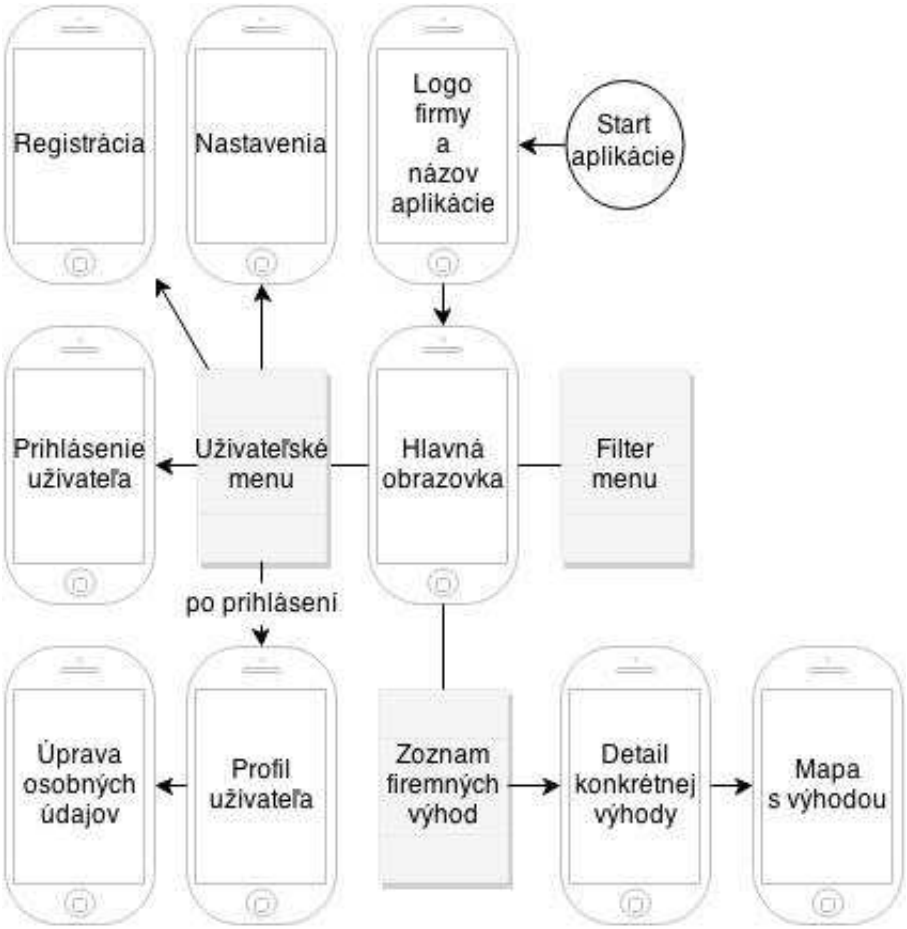
Príkladom je požiadavka aplikácie o zoznam firemných výhod. Trieda hlavnej obrazovky sa zaregistruje ako poslucháč pre udalosť firemných výhod. Keď list firemných výhod príde v odpovedi zo servera, spracuje sa a odošle pomocou konkrétnej udalosti aj s obsahujúcim listom výhod späť do hlavnej obrazovky. Hlavná obrazovka môže potom následne daný list zobraziť.



Obr. 2: Architektúra Model View Controller

6 Užívateľské rozhranie aplikácie

Aplikácia je určená pre Android, preto je užívateľské rozhranie plne ovládanteľné dotykovými gestami. Aplikácia obsahuje hlavnú obrazovku, detail výhody, obrazovku s mapou, prihlasovaciu obrazovku, registračnú obrazovku, profil užívateľa a obrazovku s úpravou osobných údajov.



Obr. 3: Zoradenie obrazoviek aplikácie

6.1 Hlavná obrazovka

Aplikácia po spustení zobrazí firemný splash screen, ktorý následne vystrieda hlavná obrazovka. Hlavná obrazovka obsahuje štandardný Java ActionBar s menu, tlačidlá filtrov pre kategórie a lokácie, zoznam všetkých firemných výhod, užívateľské menu a filter menu.

6.1.1 Užívateľské menu a filter menu

Menu obsahuje možnosti pre prihlásenie a odhlásenie používateľa a nastavenia aplikácie. Po prihlásení používateľa do tohto menu pribudne možnosť prezrieť si zakúpené výhody. Užívateľské menu môže byť vysunuté z ľavej strany obrazovky buď použitím tlačidla, ktoré sa nachádza na ActionBare, alebo dotykovým gestom "ťahanie doprava". Menu sa zasunie, keď užívateľ klikne mimo menu, alebo použije dotykové gesto "ťahanie doľava".

Menu obsahuje záložku kategórie a lokácie. V záložke kategórie sa nachádza list kategórií, z ktorých si užívateľ môže vybrať. To isté platí o záložke lokácie. Filter menu môže byť vysunuté z pravej strany obrazovky buď použitím tlačidiel filtrov pre kategórie a lokácie, alebo dotykovým gestom "ťahanie doľava". Menu sa zasunie, keď užívateľ klikne mimo menu, alebo použije dotykové gesto "ťahanie doprava". Po zavretí menu sa znovu načíta zoznam výhod podľa zvolených fitrov.

Jeden z problémov nastal v tom, že menu sú až dva, každé z jednej strany. Natívne Android metódy dokázali pracovať vždy iba s jedným menu a druhé zostalo ignorované. Zároveň som musel zamedziť aj otváraniu druhého menu počas toho, ako sa prvé zatvára.

Druhým hlavným problémom bolo, že každé menu sa otvára viacerými spôsobmi. Preto bolo treba ošetriť všetky vzájomné kombinácie otvárania oboch menu.

6.1.2 Zoznam výhod

Zoznam výhod je vždy načítaný podľa aktuálnych filtrov. Výhody sa načítajú postupne ako užívateľ scrolluje list nadol. Každá výhoda má svoj názov, obrázok, cenu a dobu platnosti. Po kliknutí na výhodu si zobrazí detail konkrétnej výhody.

Hlavný problém, ktorý sa vyskytol pri načítaní výhod, bolo potrebné nekonečné scrollovanie výhod. Musel som implementovať detekciu konca listu, ktorá následne poslala požiadavku na server o ďalšie výhody. Keď sa výhody zo servera vrátili, napojil som ich list na existujúci list výhod a prepočítal koniec listu.

6.2 Detail výhody

Detail výhody obsahuje názov, stručný popis, množstvo, obrázok a najbližšiu predajňu, v ktorej sa dá daná výhoda zakúpiť. Prihlásený užívateľ si danú výhodu môže registrovať. Každá lokácia sa po klinutí zobrazí na mape spolu s ostatnými predajňami, v ktorých sa výhoda nachádza. Taktiež sa z tohto okna dá prejsť aj do zoznamu všetkých predajní.

6.3 Obrazovka s mapou

Pokiaľ je pri konkrétnej výhode iba jedna predajňa, tak sa táto predajňa zobrazí na mape s priblížením. Pokiaľ je pri konkrétnej výhode predajní viac, budú na mape zobrazené všetky a mapa zobrazí celú Českú republiku. Taktiež sa na mape nachádza aj aktuálna pozícia zákazníka pokiaľ má v nastaveniach povolené GPS. Pri kliknutí na konkrétnu predajňu sa zobrazí názov predajne a vzdialenosť užívateľa od predajne v

kilometroch vzdušnou čiarou. Zároveň sa na spodku mapy zobrazí aj list výhod, ktoré sa v danej predajni nachádajú. Z tejto obrazovky sa dá dostať aj na natívne Google mapy.

Problém, na ktorý som natrafil pri implementácií mapy, bol fakt, že emulátor v Android Developer Kit mapy nepodoporuje a pri spustení mapy vyhadzuje výnimku s chybou. Jediným riešením bolo pripojiť reálne mobilné zariadenie k počítaču.

6.4 Prihlasovacia obrazovka

Prihlasovacia obrazovka obsahuje dve editačné polia na email a heslo, tlačidlo prihlásiť sa, registrovať sa a obnovenie strateného hesla. Každé edtačné pole má dvojitú kontrolu vstupov. Jedna kontrola prebehne, keď editačné pole stratí focus a druhá kontrola prebehne pri stlačení potvrdzovacieho tlačidla.

Problém spočíval len v ošetrení správnosti vstupov, ktoré zadáva užívateľ aplikácie. Každé pole, ktoré užívateľ môže editovať, môže obsahovať iba znaky, ktoré doň pasujú. Napríklad iba čísla v telefónnom čísle. Druhá kontrola vstupu prebieha, ked užívateľ opustí dané pole. Vtedy sa v danom poli zjaví výkričník s konkrétnym varovaním o nesprávnom vstupe. Tretia kontrola vstupov prebieha v momente, keď užívateľ stlačí potvrdzujúce tlačidlo. Pokiaľ sú v poliach nesprávne vstupy, zobrazia sa pri daných poliach varovania o chybe rovnako ako v predošlom prípade. Pokiaľ užívatel zadal správny vstup, požiadavka sa odošle na server. Server vráti správu o tom, či užívateľ zadal správne prihlasovacie údaje a podla toho užívateľ obdrží správu o jeho prihlásení.

6.5 Registračná obrazovka

Obsahuje editačné polia pre všetky potrebné údaje o zákazníkovi. Princíp kontroly vstupov je rovnaký ako v prihlasovacej obrazovke.

6.6 Profil užívateľa

Obsahuje textové polia pre všetky potrebné údaje o zákazníkovi. V ActionBare tejto obrazovky sa nachádza aj tlačidlo editácie profilu, ktoré umožní zákazníkovi meniť svoje údaje.

Pri menení užívateľských údajov platí rovnaký princíp ako pri prihlasovacej a registračnej obrazovke. Jediná zmena nastáva, že polia, ktoré užívateľ mení, neobsahujú nápovedu, ale sú už predpísané. Na potvrdenie zmeny svojich údajov musí užívateľ opätovne zadať svoje heslo.

6.7 Obrazovka s nastaveniami

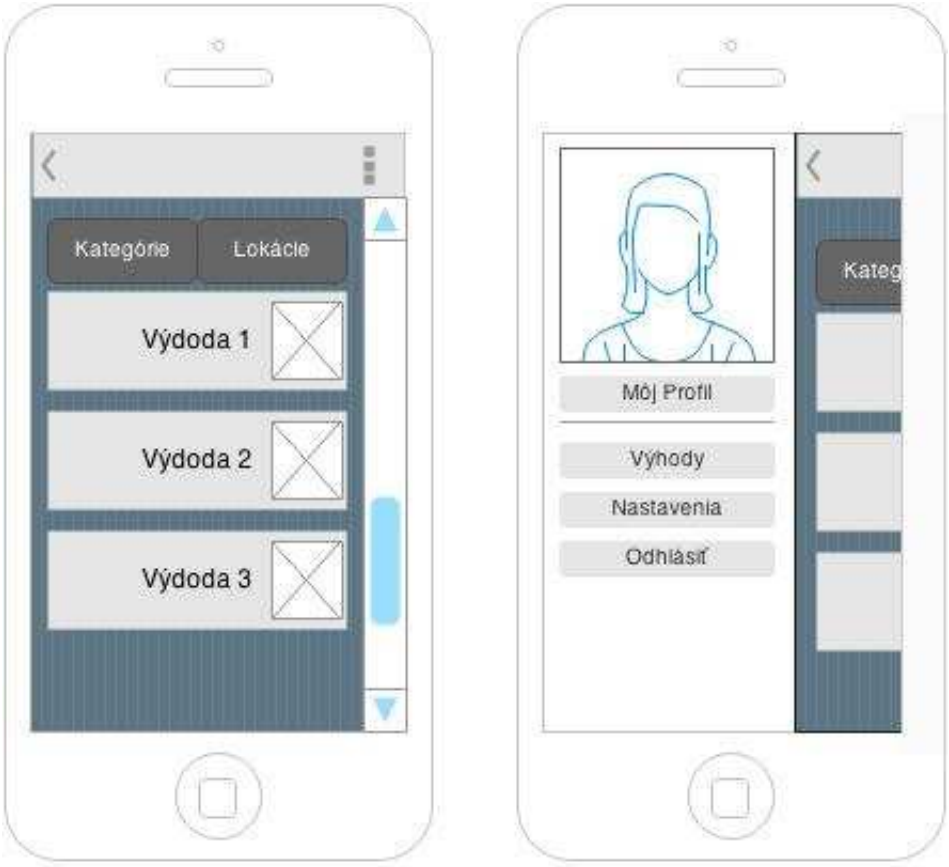
Táto jednoduchá obrazovka obsahuje možnosť zapnutia a vypnutia GPS a východzie nastavenie pre kategóriu a lokáciu.

7 Grafika užívateľského rozhrania

Finálna grafika užívateľského rozhrania ovplyvňuje hlavne vrstvu View z MVC architektúry. Preto sa do aplikácie vkladala nakoniec, keď už bolo všetko ostatné naimplemetované. Počas implementácie aplikácie som grafiku vytváral iba provizórnu, aby sa dala aplikáca ovládať. Až neskôr som potrebné grafické súbory obdržal od firemného grafika, ktorý grafiku designoval. Niektoré obrázkové súbory boli už v správnom rozlíšení, ostatné bolo treba jemne upraviť. Počas vkladania grafiky som bol v priamom kontakte s grafikom a spolu sme dolaďovali detaily. Bočné vysúvacie menu vyžadovali aj zmenu v implementácii na úrovni Controller vrstvy.

7.1 Optimalizácia pre tablety

Grafika celej aplikácie musela byť optimalizovaná pre tablety. Táto časť ovplyvňuje čisto len vrstvu View. Každý Android projekt obsahuje rozmerové súbory pre tri rozlíšenia. Tie som musel editovať osobitne pre každé zariadenie.



Obr. 4: Ukážka hlavnej obrazovky a užívateľského menu

8 Porovnanie práce na praxi a v škole

8.1 Práca v kolektíve

Práca v kolektíve tejto firmy sa mi veľmi páčila. Ako prvý som si ale všimol rozdiel medzi programovaním vo firme a programovaním skupinového projektu v škole. Problém spočíva hlavne v rozdelení práce. Vo firme má každý zamestnanec svoju časť projektu, ktorú vytvára sám a túto časť postupne napája na ostatné časti, ktoré vytvárajú kolegovia. Keď sa však robil skupinový projekt v škole, tak dostalo viac žiakov jeden nerozdelený celok. Vrámci zadania si síce mali medzi sebou rozdeliť úlohy, ale v realite to dopadlo väčšinou tak, že celý projekt vypracoval len jeden z nich.

Vo firme sme na celkovom projekte pracovali piati. Môj mentor mal za úlohu riešiť logiku alikácie a zároveň implementáciu tej istej aplikácie na operačný systém iOS. Ja som túto logiku aplikácie implementoval na operačný systém Android a ďalší praxujúci spolužiak vyvíjal aplikáciu pre Windows Phone. Aplikácia na svoju bežnú funkčnosť potrebuje komunikovať s webovým serverom. O túto časť sa staral ďalší zamestanec. Počas toho ako každý implementoval svoju časť, grafik designoval vizuálnu stránku aplikácie, pričom nám priebežne dával vedieť zmeny, ktoré ovplyvňovali aj logiku aplikácie. Potom sa táto grafika napojila na naimplementovanú mobilnú aplikáciu a doladili sa posledné detaily.

8.2 Štruktúra zdrojového kódu[4]

Hoci programátorské normy ako napríklad malé a veľké písmená pri názvoch tried, premenných a metód, sú všade rovnaké, v zdrojovom kóde dbala firma aj na iné pravidlá. Hlavným pravidlom bolo dodržanie prehľadnosti kódu. Preto musela mať každá metóda komentár popisujúci správanie metódy ako aj jej vstupné parametre a návratové hodnoty. Ďalšou dôležitou podmienkou bolo riadkovanie kódu. Pokiaľ to bolo možné, tak sa riadky v metóde delili medzi inicializáciu premenných, napĺňanie premenných a prácu s premennými.

Záver

Z tejto praxe som si veľa odniesol. Celkovo ju hodnotím pozitívne a som rád, že som si ju vybral. Práca bola veľmi zaujímavá a dynamická. Svoje programátorské schopnosti som počas tohto obdobia nielen otestoval, ale aj obohatil. Taktiež som si vyskúšal prácu v kolektíve, pri ktorej som zistil, ako to naozaj funguje v zamestnaní. Myslím si, že školská príprava na takúto prácu je skvelá, aj keď niektoré pracovné postupy som sa musel preúčať. Každopádne si ale viem reálne predstaviť, že by som sa takouto prácou v budúcnosti živil.

Peter Slivoš

9 Literatúra

[1] Design Patterns: Elements of Reusable Object-Oriented Software. USA: Addison-Wesley, 1994. ISBN 0-201-63361-2.

[2] LEE, Wei-Meng H. Beginning android 4 application development. 1st ed. Indianapolis, IN: Wiley Pub., Inc., 2012. ISBN 11-181-9954-5.

[3] BLOCH, Joshua. Effective java. 2nd ed. Upper Saddle River: Addison-Wesley, c2008, xxi, 346 s. ISBN 0321356683.

[4] MARTIN, Robert C. Clean code: a handbook of agile software craftsmanship. Upper Saddle River, NJ: Prentice Hall, c2009. ISBN 01-323-5088-2.